

Rapport de Projet Traitement d'images

Marion Pavaux

21 Mai 2021

1 Déroulement des opérations

Le but du projet était de trouver la hauteur des amas blancs qui se déplacent le long du tube au centre de l'image, et de les segmenter. Pour ce faire, la vidéo a été chargée image par image, et chacune d'elle a été modifiée pour permettre les opérations de segmentation. Une hauteur plus grande de pixels blancs sur l'image indique la présence d'un slug. Globalement, la méthode de segmentation utilisée a été de compter le nombre de pixels blancs par colonne. La vidéo a d'abord été rognée, puis débruitée et mise en noir et blanc pour enfin permettre le début des étapes de segmentation.

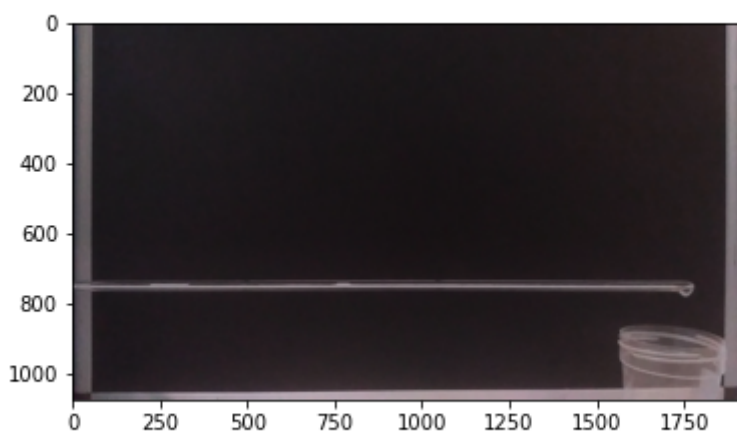


Figure 1: Une image non traitée issue de la vidéo

1.1 Encadrer la zone d'intérêt

Au début de la vidéo, une main installe l'expérience. Il faut donc exclure cette partie de la vidéo qui brouillerait le programme, après les premières 20s de la vidéo, soit à environ la 600^{ieme} image. Toute l'image n'est pas d'intérêt pour segmenter, il faut établir

une fenêtre qui ne prend en compte que la ligne où les amas blancs se déplacent. Aussi l'image étant faussement en niveaux de gris, seul le premier canal a été retenu afin de faciliter le traitement.

```
frame = frame_ini[600:900,75:1700,0]
```

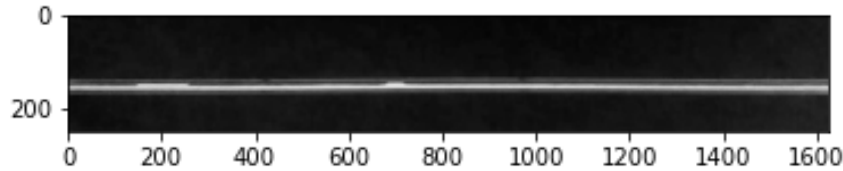


Figure 2: Fenêtre de travail de l'image

1.2 Débruiter l'image

Pour débruiter l'image, un simple filtre médian de la bibliothèque `ndimage` a été utilisé. Il semble que c'était le filtre marchant le mieux avec notre problème. Une taille de 2 ou de plus de 3 diminuait moins le bruit de un pixel.

```
imconv = ndimage.median_filter(image,(3,3))
```

Voici le résultat:

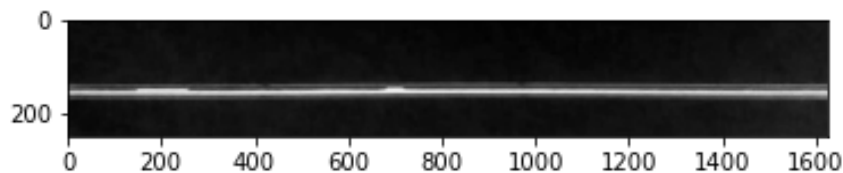


Figure 3: Frame après débruitage

1.3 Thresholding

Pour trouver la hauteur des amas blancs, un comptage des pixels blancs va être réalisé. Nous avons donc besoin de transformer les pixels clairs en blancs et foncés en noirs. Cela se fait à l'aide de la fonction `thresholding`. Elle crée une nouvelle image qui, si le pixel de l'image initiale a une valeur supérieure au seuil de blancheur, le rend complètement blanc dans la nouvelle image, et noir sinon.

Sur le deuxième exemple, le seuil n'a pas été correctement choisi, entraînant du bruit blanc dans l'image.

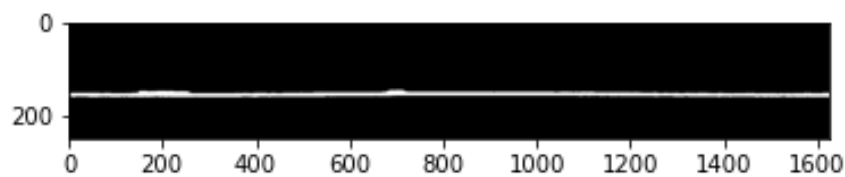


Figure 4: Frame après l'opération thresholding - seuil=120

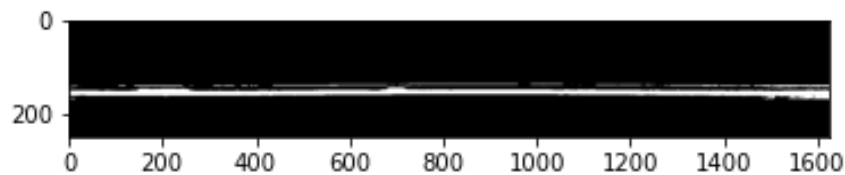


Figure 5: Frame après l'opération thresholding - seuil=80

1.4 Trouver le nombre de pixels blancs par colonne de pixels

Dans notre cas savoir s'il y a un slug revient à savoir si le nombre des pixels blancs est inhabituel par colonne.

Sur l'image de pixels noirs et blancs, la fonction comptage part du bas de chaque colonne, et commence à compter si elle voit un pixel blanc et s'arrête lorsque qu'un pixel redevient noir. Elle range ensuite ces nombres dans un tableau de la taille de la largeur de l'image, et la hauteur d'arrêt des pixels blancs dans un autre tableau.

Ci-dessous, la représentation graphique de ce tableau de comptage. On voit clairement qu'il reste du bruit, mais celui-ci ne sera pas gênant dans la détection finale, étant toujours de un pixel.

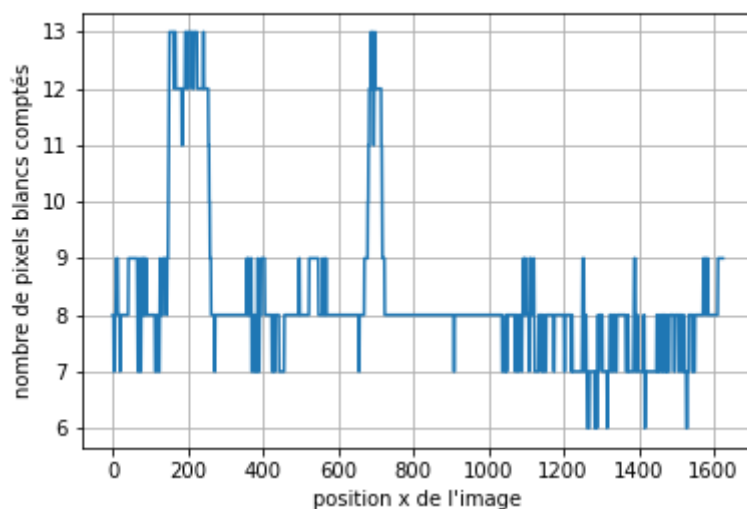


Figure 6: Nombre de pixels blancs comptés par colonne d'image

1.5 Trouver la hauteur et la position des slugs

Pour trouver la hauteur des slugs, la fonction `compute_features` prend en argument, le tableau de comptage précédent, la hauteur de fin de pixels blancs par colonne et l'image à traiter. Elle renvoie un dictionnaire de positions qui contient, pour chaque slug, la colonne de début, de fin, la moyenne sur sa longueur de son épaisseur, et la moyenne sur sa longueur de sa hauteur. Les positions, et hauteurs, selon x et y sont données par rapport à l'image initiale et non plus par rapport à l'image rognée. On a donc ajouté selon les rognages de départ, 75 pixels selon x et 600 selon y.

La fonction opère de la façon suivante: elle trouve le maximum et l'écart type du tableau de comptage. Des valeurs typiques étaient $m = 13$ pour le maximum, et $\sigma = 1.37$ pour l'écart type. Elle considère qu'elle trouve le début d'un slug lorsque le comptage est supérieur au maximum moins deux fois l'écart type, ce qui permet en même temps de ne pas être dérangés par le bruit restant.

```
if maxi - 2*sigma <= feature[ind]:
```


Elle note alors le numéro de colonne dans la variable `begin` et elle continue, jusqu'à ce qu'elle ne trouve pas le comptage suffisant. Elle note alors le numéro de colonne dans la variable `end`.

Voici le résultat de `compute_features` sur les images données ci-dessus. Elle trouve bien deux slugs, un premier long et un deuxième plus court.

```
[[224, 332, 3.3119266055045866, 769.8256880733945]  
[756, 792, 2.9459459459459456, 767.3243243243244]]
```

2 Résultats

Mon ordinateur a mis une minute trente à raison du traitement d'une image sur six à partir de la vingtième seconde de la vidéo. Dans le terminal, l'appel à la fonction `main` nous donne:



```
L'épaisseur moyenne des slugs est:  
10.97459967370041  
La hauteur moyenne des slugs est:  
759.4295023742045
```

Figure 7: Résultats de l'appel à la fonction `main`

Ces moyennes sont les moyennes effectuées sur chaque image, et on peut considérer que chaque slug apparaissant environ le même nombre de fois dans les images, apporte une contribution égale à la moyenne.

Un traitement supplémentaire a été fourni afin de colorier, sur l'image initiale les slugs en vert sur une épaisseur de quatre pixels en haut et quatre pixels en bas par rapport à leur hauteurs respectives. Les images ont ensuite été ajoutées à un tableau, et traitées toutes à la fin à l'aide de `cv2.VideoWriter` pour former une vidéo de 6 fps. Je l'ai laissée dans le dossier de rendu.

Enfin, la fonction retourne le tableau, des résultats de `find_position` pour chaque frame traitée.

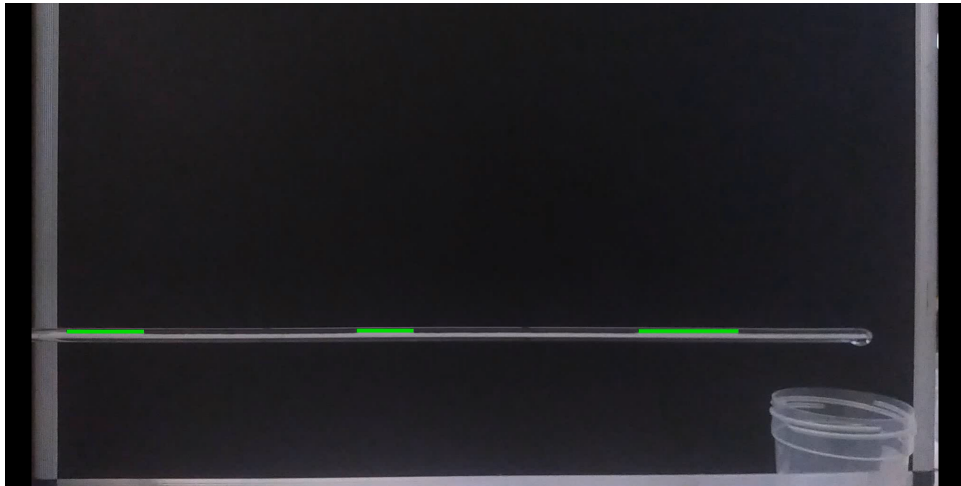


Figure 8: Image de la vidéo de sortie, représentant en vert les slugs